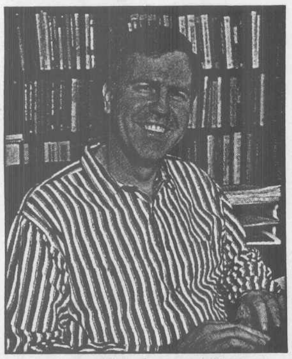
EDN COLUMNIST



DAVID BRUBAKER, FUZZY-LOGIC CONTRIBUTING EDITOR

PHOTOGRAPHY / LIANE ENKELIS

Smooth thresholds using Zadehan logic

A friend recently told me he felt that most fuzzy proponents view fuzzy systems as "*fuzzy* systems." When he said the word "fuzzy," he held his hands wide apart, as if describing the latest fish he had caught, and when he said "systems," he held his thumb and forefinger so they nearly touched. Although I agree for the most part, I informed him that one could design a system that uses fuzzy, or Zadehan, logic without making it the dominant technology. Here is an example of how I inserted a small fuzzy component into an otherwise nonfuzzy system.

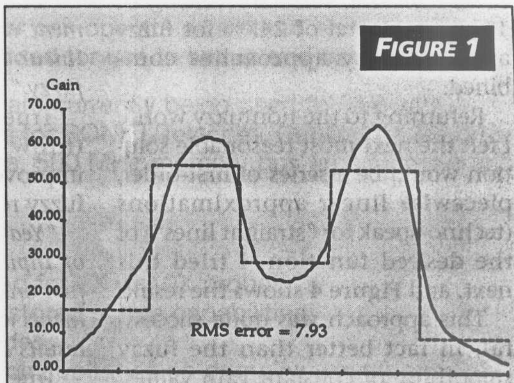
A client company had a proprietary algorithm it used in a real-time analysis system. The company had originally developed the algorithm in the mid-1970s and continued to improve it over the next two decades. For quite a few of those years, however, the employees had not touched the core of the algorithm, primarily because its author had retired. In addition, none of the employees had the time to learn the core's structure well enough to safely modify it. They contracted me to perform this task. A number of the engineers at the company knew I worked with fuzzy systems, and, on the first day, the program manager made it clear that he wanted nothing fuzzy in my modifications.

I will describe only one of several modifications I made to the algorithm. It was implemented in a section that calculated a gain parameter based on the value of an input variable.

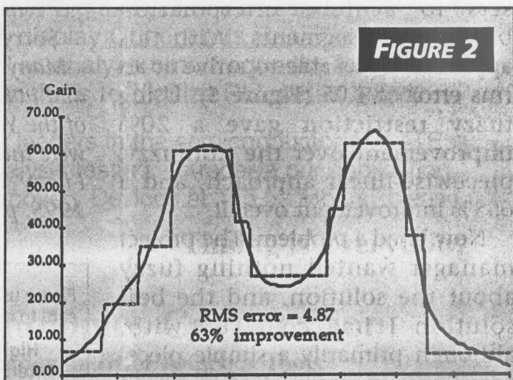
The original design divided the input's range into five equal-width regions, with a constant gain value selected for each region. Over the years, a great deal of data had been gathered that indicated what this gain function should be, and although the employees had changed the constant gain values accordingly, they had not modified the thresholds that divided the input range into the five regions. **Figure 1** shows both the desired gain function, derived from the data, and the approximation, as the algorithm calculated it when I first joined the project.

As a measure of how accurately the calculated function approximated the desired function, I used the rms value of the error, where the error was the point-by-point difference between the desired and approximated values. The starting point for the original five-segment version was an rms-error value of 7.93.

The first improvement was a no-brainer. The processor being used was fast enough and the memory plentiful enough that I could easily double the number of regions.



The smooth plot results from reducing gain-data gathered over many years of operation. The dashed line shows the starting-point approximation, using five equal-width regions and constant gain values within those regions. The rms error of the approximation is 7.93.



The first improvement results from using 10 regions of varying widths, although there are still constant gain values within each region. With an rms error of 4.87, there is a 63% improvement over the approximation in Figure 1.

Figure 2 shows that using 10 regions, each with constant gain, helps approximate the desired gain. Note that the region widths are no longer identical. The rms error of this method is 4.87, or a 63% improvement—that is, the new version is a factor 1.63 better than the original.

The next step I took covertly: I wanted to see what further improvement a fuzzy approximator would bring. With the same 10 input segments and the same output values within the segment, I used overlapping fuzzy sets to interpolate between the constant value regions (**Figure 3**). The rms error dropped further down to 2.28, an improvement over the nonfuzzy version of 113%, or a total of 247% for fuzzy and nonfuzzy approaches combined.

Returning to the nonfuzzy world, I felt the next most reasonable solution would be a series of first-order, piecewise linear approximations (techno-speak for "straight lines") of the desired function. I tried this next, and **Figure 4** shows the result.

This approach was quite successful, in fact better than the fuzzy smoothing of constant gain values, with an rms error of 1.26, or a whopping 529% overall improvement. Again, out of curiosity and acting undercover, I added fuzzy restriction, which qualifies (multiplies) each straight-line equation with a fuzzy set and overlaps adjacent fuzzy sets to achieve interpolation between the segments. With this approach, I was able to arrive at an rms error of 1.05 (**Figure 5**). Using fuzzy restriction gave a 20% improvement over the nonfuzzy, piecewise-linear approach, and a 655% improvement overall.

Now I had a problem. The project manager wanted nothing fuzzy about the solution, and the best solution I had come up with, although primarily a simple piecewise-linear approximation, did have a fuzzy component. Did I sell him a solution or a technology (or rather, in this case, a nontechnology)? After half a second of deep deliberation, I installed the fuzzy approach. In the design review, with straight face and serious demeanor and never mentioning the word "fuzzy," I explained in great detail what I had done. Using a piecewise-linear approximator occurred to most of the reviewers before I got to it, and they especially appreciated the way I had achieved smooth transitions among the various segments. End of case study.

I can already hear your "Yeah, buts..."

"Yeah, but that's not really fuzzy logic. Where are the rules?"

There is nothing that says a fuzzy-logic system needs to be rule based.

"Yeah, but most of the improvement was a result of the piecewise-linear approach—not that it was fuzzy."

True, but are you willing to throw away the final 20% improvement that the addition of fuzzy restriction achieved?

"Yeah, but doubling the number of input regions again and using first-order, piecewise-linear line segments would have achieved the same results as your fuzzy restriction."

True, I have never claimed other approaches could not achieve the same results as fuzzy logic, especially in simple cases like this one. Besides, I could apply fuzzy restriction to any number of piecewise-linear segments and achieve a further reduction in error.

"Yeah, but..."

Sorry—we're out of room.

Many thanks to those who noticed and notified me that the definitions of the Yager AND and OR operators were interchanged in the column on Fuzzy Operators (EDN, Nov 9, 1995, pg 240).

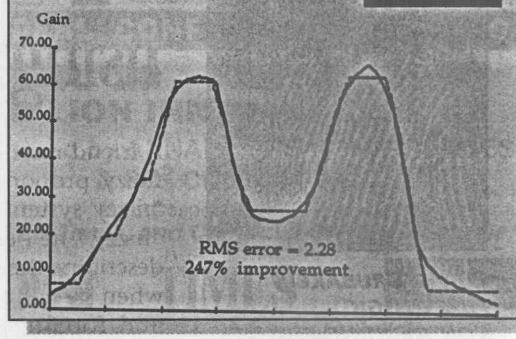
VOTE

Please use the Information Retrieval Service card to rate this article (circle one):

High Interest	Medium Interest	Low Interest
594	595	596

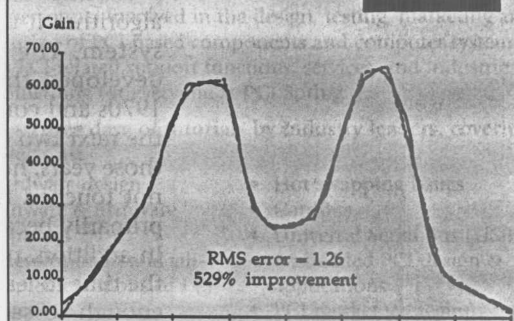
Dr David Brubaker is a consultant in fuzzy-system design. You can reach him at brubaker@ cup.portal.com.

FIGURE 3



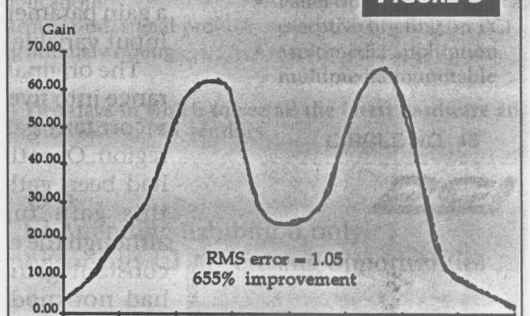
Using fuzzy logic to smooth the steps in Figure 2 causes the rms error to drop to 2.28, with a 247% improvement over the starting point in Figure 1.

FIGURE 4



Replacing constant-value gains with first-order, piecewise-linear gain functions in each region and slightly adjusting the region boundaries results in an rms error of 1.26, a 529% overall improvement.

FIGURE 5



Using fuzzy restriction to smooth the region transitions of the piecewise-linear approach results in an rms error of 1.05, an overall improvement of 655% from the first approximation in Figure 1.